A Parallel Tabu Search Strategy for Cell Placement in VLSI Circuit Design

Sadiq M. Sait Sanaullah Syed

College of Computer Sciences & Engineering
King Fahd University of Petroleum & Minerals
Dhahran 31261, Saudi Arabia
E-mail: {sadiq,sanaulla} @kfupm.edu.sa

***Abstract*— Tabu Search based cell placement approaches for VLSI circuit design have shown excellent results when executed on a single processor. However, they require significant computation time. Of the various acceleration strategies attempted, parallelization has always exhibited the most potential. The parallel Tabu Search approach presented in this work can be classified as a synchronous master-slave p-control, RS and MPSS strategy. The approach is implemented on a dedicated Linux-based cluster of workstations, using MPI libraries for communication. Experimental results for ISCAS'89 benchmark circuits show excellent performance in terms of scalability & speed-up.**

I. Introduction & Motivation

Due to its complexity, the VLSI (Very Large Scale Integration) design process is divided into several intermediate levels of abstraction. In this work, optimization at the physical level, in particular the cell placement phase [1], is of concern. Basically, placement in the process of arranging various circuit components on a layout surface. Achieving such multi objective optimization with the sheer complexity of modern circuit density is an NP-hard problem for which conventional constructive techniques have often proved inadequate. Iterative heuristics on the other hand have been a tremendous success in reaching acceptable solutions for such problems. The primary advantage of iterative heuristics over conventional constructive algorithms is their probabilistic ability to escape from local optima.

Tabu Search heuristic is based on the systematic exploration of memory functions. It is an aggressive search technique where for a given solution, a large number of neighbors are generated, from which the best is chosen. To determine which of the generated solutions is the best, an *evaluator* that is based on the objectives being optimized, and the historical information that has been accumulated, is used. The trace of the current solution is controlled by a recent move history to avoid cycling in the solution space. Use of intensification/diversification in Tabu Search considerably helps in obtaining superior quality solutions. This is accomplished with the help of additional memory structures that keep record of information such as frequencies of moves, elite solutions, etc. An algorithmic description of a short-term Tabu Search is given in [2], [3].

Despite the advances in VLSI technology, there are still a few challenges that pose an obstacle in its rapid development. Almost all the steps in VLSI CAD applications, like synthesis, analysis and verification take large run-times on single machines even with iterative heuristics. Of the various acceleration strategies attempted, parallel computing has always exhibited the most potential. Not only is it possible to achieve shorter run-times with parallel processing but also handle larger problem sizes and obtain better quality results by traversing larger search spaces [4], [5].

This paper is organized as follows: The following section reviews some previous efforts for parallelizing Tabu Search. In Section III, the proposed parallel Tabu Search approach is presented, followed by experimental setup in section V. Experimental results and discussions are reported in section VI.

II. Related work

A generic intuitive strategy for achieving parallelization is to partition the data into small subsets distributed among the processors [6], [7]. Each processor is responsible for a data subset and implements a sequential version of the concerned heuristic over this data subset.

However, for combinatorial optimizations, three types of parallelization strategies seem to be appropriate [8]. They are

1) The operations within an iteration of the solution method are parallelized.
2) The search space (problem domain) is decomposed.
3) Multi-search threads with various degrees of synchronization are used.

A taxonomy of Parallel Tabu Search strategies was given by Crainic et. al [9]. The authors classify parallel Tabu Search strategies along three dimensions. The first dimension is *Control cardinality*, where the strategy is classified either as *1-control* or *p-control*. The second dimension is *Control and communication type*, where the strategy can follow a *rigid synchronization (RS)*, *knowledge synchronization (KS)*, *Collegial (C)*, or a *Knowledge Collegial (KC)* strategy. The third dimension is *Search differentiation* where the strategy can be *SPSS (single point single strategy)*, *SPDS (single point different strategies)*, *MPSS (multiple point single strategy)*, or *MPDS (multiple point different strategies)*.

The first reported studies on the parallelization of Tabu Search were published in the early 1990s [10], [11], [12]. Based on the above taxonomy, the previous Tabu Search algorithms have been categorized by [9]. In order to solve the flow shop sequencing problem by Taillard [10], a mechanism for parallel implementation of Tabu Search algorithm used search space decomposition strategy. It is a *1-control, RS, SPSS algorithm*. Another parallelization of Tabu Search for vehicle routing problem by Garica et. al [11] also uses search space decomposition strategy. It is also *1-control, RS, SPSS* algorithm. In order to improve parallel Tabu Search using evolutionary principles, the algorithm presented by Falco et. al [12] used *multi-search thread* strategy. It is a *p-control, C, MPSS* algorithm. Another parallel Tabu Search algorithm for the 0-1 multidimensional knapsack problem was put forth by Nair et. al [13], which used *multi-search threads* strategy. It is a *p-control, RS, MPDS* algorithm. In [14], a parallel Tabu Search algorithm for voltage and reactive power control in power systems was proposed. Of the two schemes, one of them used the *domain decomposition strategy*, while the other scheme followed a *multi-search threads* strategy. The first one is *1-control, RS, SPSS* and the second one is *p-control, RS, MPDS* algorithm.

III. Parallel Tabu Search for Placement

The parallel Tabu Search approach presented in this work can be classified as a synchronous master-slave (one master and remaining slaves), p-control (each process is responsible for its search), rigid synchronization (RS) (synchronous operation mode where processes are forced to establish communication and exchange information at specific points explicitly defined) and Multiple Point Single Strategy (MPSS) (processes start with different initial solutions but all follow the same strategy). In this implementation, a master process runs on one machine and the slave processes runs on distinct machines. All processes start with the different initial solutions. After searching its local neighborhood for some fixed number of iterations (100 in our case), each slave process reports its best solution back to the master. The local neighborhood is obtained by dividing the actual neighborhood size on a single machine, among the slave processors. The slave processes maintain their own Tabu lists and apply their aspiration criteria. The master process selects the overall best among the received best solutions subject to Tabu conditions. If the stopping criteria are met then the search stops; otherwise the master broadcasts the selected solution back to the slaves and the search continues.

IV. The Placement Problem and Cost Functions

A. VLSI Standard Cell Placement

The cell placement problem can be stated as follows: Given a collection of cells or modules with ports (inputs, outputs, power and ground pins) on the boundaries, the dimensions of these cells (height, width, etc), and a collection of nets (which are sets of ports that are to be wired together), the process of *placement* consists of finding suitable physical locations for each cell on the entire layout. By suitable we mean those locations that minimize given objective functions, subject to certain constraints imposed by the designer, the implementation process, or layout strategy and the design style. The cells may be standard-cells, macro blocks, etc.

In this work, we deal with standard cell design, where all the circuit cells are constrained to have the same height, while the width of the cell is variable and depends upon its complexity. The three design objectives considered are optimization of power dissipation, delay and wire length. In standard CMOS technology, power dissipation is a function of the clocking frequency, supply voltages and the capacitances in the circuit. The delay of any given path is computed as the summation of the delays of the nets $v_1, v_2, ..., v_k$ belonging to that path and the switching delay of the cells driving these nets. Steiner tree approximation is computed for each net and the summation of all Steiner trees is considered as the interconnection length of the proposed solution.

In standard cell placement, cells (or blocks) of fixed heights are placed in rows. It is the width of these rows that varies with the proposed solution according to the type and number of cells placed in the row. An approximation would be to treat cells as points, but in order to estimate lengths of interconnects more accurately, widths of cells are taken into account. Heights of routing channels are estimated using the vertical constraint graphs constructed during the channel routing phase. With this information, a fairly accurate estimate of power dissipation, delay and total wire length can be obtained [1].

B. Cost Functions

Now we formulate cost functions for our three said objectives and for the width constraint.

1) Wire length Cost:: Interconnect Wire length of each net in the circuit is estimated and then total wire length is computed by adding all these individual estimates:

$$Cost_{wire} = \sum_{i \in M} l_i \tag{1}$$

where l_i is the wire length estimation for net i and M denotes total number of nets in circuit (which is the same as number of modules for single output cells).

2) Power Cost:: Power consumption p_i of a net i in a circuit can be given as:

$$p_i \simeq \frac{1}{2} \cdot C_i \cdot V_{DD}^2 \cdot f \cdot S_i \cdot \beta \tag{2}$$

where C_i is total capacitance of net i, V_{DD} is the supply voltage, f is the clock frequency, S_i is the switching probability of net i, and β is a technology dependent constant.

Assuming a fix supply voltage and clock frequency, the above equation reduces to the following:

$$p_i \simeq C_i \cdot S_i \tag{3}$$

The capacitance C_i of cell i is given as:

$$C_i = C_i^r + \sum_{j \in M_i} C_j^g \tag{4}$$

where C_j^g is the input capacitance of gate j and C_i^r is the interconnect capacitance at the output node of cell i.

At the placement phase, only the interconnect capacitance C_i^r can be manipulated while C_j^g comes from the properties of the cell library used and is thus independent of placement. Moreover, C_i^r depends on wire length of net i, so equation 3 can be written as:

$$p_i \simeq l_i \cdot S_i \tag{5}$$

The cost function for total power consumption in the circuit can be given as:

$$Cost_{power} = \sum_{i \in M} p_i = \sum_{i \in M} (l_i \cdot S_i) \tag{6}$$

3) Delay Cost:: Delay cost is determined by the delay along the longest path in a circuit. The delay T_π of a path π consisting of nets $\{v_1, v_2, ..., v_k\}$, is expressed as:

$$T_\pi = \sum_{i=1}^{k-1} (CD_i + ID_i) \tag{7}$$

where CD_i is the switching delay of the cell driving net v_i and ID_i is the interconnect delay of net v_i. The placement phase affects ID_i because CD_i is technology dependent parameter and is independent of placement.

The delay cost function can be written as:

$$Cost_{delay} = max\{T_\pi\} \tag{8}$$

4) Width Cost:: Width cost is given by the maximum of all the row widths in the layout. We have constrained layout width not to exceed a certain positive ratio α to the average row width w_{avg}, where w_{avg} is the minimum possible layout width obtained by dividing the total width of all the cells in the layout by the number of rows in the layout. Formally, we can express width constraint as below:

$$Width - w_{avg} \leq \alpha \times w_{avg} \tag{9}$$

5) Overall Fuzzy Cost Function:: Since, we are optimizing three objectives simultaneously, we need to have a cost function that represents the effect of all three objectives in form of a single quantity. We propose the use of fuzzy logic to integrate these multiple, possibly conflicting objectives into a scalar cost function. Fuzzy logic allows us to describe the objectives in terms of linguistic variables. Then, fuzzy rules are used to find the overall cost of a placement solution [15]. In this work, we have used following fuzzy rule:

IF a solution has
SMALL wire length **AND**
LOW power consumption **AND**
SHORT delay
THEN it is an *GOOD* solution.

The above rule is translated to *and-like* OWA fuzzy operator [16] and the membership $\mu(x)$ of a solution x in fuzzy set *GOOD solution* is given as:

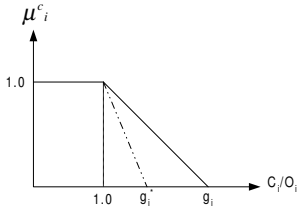

Fig. 1. Membership functions

$$\mu(x) = \begin{cases} \beta \cdot \min\limits_{j=p,d,l} \{\mu_j(x)\} + (1-\beta) \cdot \frac{1}{3} \sum\limits_{j=p,d,l} \mu_j(x); \\ \qquad \text{if} \quad Width - w_{avg} \leq \alpha \cdot w_{avg}, \\ 0; \qquad \text{otherwise.} \end{cases} \tag{10}$$

Here $\mu_j(x)$ for $j = p, d, l, width$ are the membership values in the fuzzy sets *LOW power consumption, SHORT delay*, and *SMALL wire length* respectively. β is the constant in the range $[0, 1]$. The solution that results in maximum value of $\mu(x)$ is reported as the best solution found by the search heuristic.

The membership functions for fuzzy sets *LOW power consumption, SHORT delay*, and *SMALL wire length* are shown in Figure 1. We can vary the preference of an objective j in overall membership function by changing the value of μ_j . The lower bounds O_j for different objectives are computed as given in Equations 11-14:

$$O_l = \sum_{i=1}^{n} l_i^* \quad \forall v_i \in \{v_1, v_2, ..., v_n\} \tag{11}$$

$$O_p = \sum_{i=1}^{n} S_i l_i^* \quad \forall v_i \in \{v_1, v_2, ..., v_n\} \tag{12}$$

$$O_d = \sum_{j=1}^{k} CD_j + ID_j^* \quad \forall v_j \in \{v_1, v_2, ..., v_k\} \; in \; path \; \pi_c \tag{13}$$

$$O_{width} = \frac{\sum_{i=1}^{n} Width_i}{\#\ of\ rows\ in\ layout} \tag{14}$$

where O_j for $j \in \{l, p, d, width\}$ are the optimal costs for wire-length, power, delay and layout width respectively, n is the number of nets in layout, l_i^* is the optimal wire-length of net v_i, CD_i is the switching delay of the cell i driving net v_i, ID_i is the optimal interconnect delay of net v_i calculated with the help of l_i, S_i is the switching probability of net v_i, π_c is the most critical path with respect to optimal interconnect delays, k is the number of nets in π_c and $Width_i$ is the width of the individual cell driving net v_i.

V. Experimental setup

The experimental setup consists of the a homogenous cluster of 7 machines, x86 architecture, Pentium-4 of 2 GHz clock speed, and 256 MB of memory. These machines are connected by 100Mbit/s ethernet switch. Operating system used in Linux

7.3 (kernel 2.4.7-10). The paradigm for parallel environment used is MPI (Message Passing Interface). MPICH, a portable implementation of MPI standard 1.1 is used. In terms of GFlops, the maximum performance of the cluster, with NAS Parallel Benchmarks was found out to be 1.6 GFlops, (using NAS's LU, Class A, for 8 processors). Using this same benchmark for a single processor, the individual performance of one machine was found out to be 0.3 GFlops. The maximum bandwidth that was achieved using PMB was 91.12 Mbits/sec, with a latency of 68.69 μsec per message.

VI. Results and Discussion

In this work, ISCAS'89 benchmarks circuits are used. These contain a set of circuits with various sizes, in terms of number of gates and paths. The results of various circuits using a sequential Tabu Search [17] on a single processor (a single machine from the cluster) are tabulated in Table I. Here 'WL', 'P' and 'D' are the wire length, power and delay costs respectively, whereas the aggregate fuzzy cost is denoted by (μ) and 'T' is the execution time in seconds.

TABLE I
Results for ISCAS'89 circuits for one processor

Circuit	Gates	Paths	WL	P	D	μ	T
s298	136	150	4545	863	127	0.726	56
s386	172	205	6520	1582	188	0.683	104
s641	433	687	12176	2834	659	0.799	1865
s832	310	240	17878	4016	355	0.651	160
s953	440	583	25771	4041	202	0.670	391
s1196	561	600	35690	10777	316	0.676	752
s1238	540	661	38913	11473	358	0.639	755
s1488	667	557	54786	13670	651	0.620	538
s1494	661	558	52209	12880	565	0.631	541
c3540	1753	668	168831	59724	695	0.693	3843
s9234	5844	512	938313	170119	969	0.689	10717
s15850	10470	512	2921966	228512	1831	0.683	20192

We now report the results obtained from the proposed parallel Tabu Search heuristic. Table II shows the execution times as well as the corresponding speed-ups for 2, 4, and 6 processors. As can be seen, there is almost linear speed-up in most of the cases with minor variations. A significant observation is that in case of larger circuits, excellent results are obtained in terms of scalability and speed-up. For instance, in case of s15850 having 10470 gates, the execution time is reduced from 20,192 seconds down to 3,441 seconds when using 6 processors resulting in speed-up of 6.03.

As mentioned above in Section III, the neighborhood size is divided among the slave processors. Since the number of neighbor solutions generated at each slave is reduced with the increasing number of processors, the quality of the solution decreases. The reason for this degradation is that each slave has a lower number of possible moves to make and hence search pool is reduced. It is observed that the percentage of quality degradation ranges from 5% in case of larger circuits to 10% in case of smaller circuits as compared to that obtained by sequential Tabu Search strategy.

TABLE II
Runtimes (sec) and speedups of proposed parallel Tabu Search on 2,4 and 6 processors versus Tabu Search on a Single processor.

Circuit	Proc=1	Proc = 2	Proc = 4	Proc = 6
s298	56	34 (1.65)	19 (2.95)	15 (3.73)
s386	104	56 (1.86)	29 (3.58)	21 (5.47)
s641	1865	962 (1.94)	503 (3.71)	308 (6.05)
s832	160	87 (1.83)	44 (3.64)	26 (6.15))
s953	391	201 (1.95)	104 (3.76)	62 (6.30))
s1196	752	401 (1.87)	200 (3.76)	130 (5.78)
s1238	755	390 (1.93)	199 (3.79)	123 (6.14)
s1488	538	283 (1.90)	144 (3.73)	93 (5.78)
s1494	541	287 (1.88)	144 (3.75)	93 (5.81)
c3540	3843	2020 (1.90)	986 (3.89)	621 (6.18)
s9234	10717	5547 (1.93)	2759 (3.88)	1777 (6.03)
s15850	20192	10621 (1.90)	5294 (3.81)	3441 (6.07)

VII. Conclusions

In this work, we presented a parallel Tabu Search strategy for VLSI standard cell placement. The proposed strategy belongs to p-control, RS, MPSS class. The experimental results exhibit excellent scalability and almost linear speed-ups as the number of processors increase. However, we observed a trade-off between the reduction in runtime and the quality of placement solution. In most cases, a speed-up of 5 to 6 was obtained at the cost of 5-10% degradation in quality.

VIII. Acknowledgement

The authors would like to thank King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia, for support under project code # COE/CELL PLACE/263.

References

[1] Sadiq M. Sait and Habib Youssef. *VLSI Physical Design Automation: Theory and Practice*. World Scientific, Singapore, 2001.

[2] F. Glover. Tabu search: A tutorial. *Technical Report, University of Colorado, Boulder*, February 1990.

[3] Sadiq M. Sait and Habib Youssef. *Iterative Computer Algorithms and their Application to Engineering*. IEEE Computer Society Press, December 1999.

[4] Prithviraj Banerjee. *Parallel Algorithms for VLSI Computer-Aided Design*. Prentice Hall International, 1994.

[5] Van-Dat Cung, Simone L. Martins, Celso C. Riberio, and Catherine Roucairol. Strategies for the Parallel Implementation of metaheuristics. *Essays and Surveys in Metaheuristics*, pages 263–308, Kluwer 2001.

[6] A. Casotto, F. Romeo, and A. L. Sangiovanni-Vincentelli. A parallel simulated annealing algorithm for the placement of macro-cells. *IEEE Transactions on Computer-Aided Design*, CAD-6(5):838–847, September 1987.

[7] P. Banerjee and M. Jones. A parallel simulated annealing algorithm for standard-cell placement on a hypercube computer. *Proceedings of International Conference on Computer-Aided Design, ICCAD-86*, 1986.

[8] M. Toulouse, T. G. Crainic, and M. Gendreau. Issues in Designing Parallel and Distributed search Algorithms for Discrete Optimization Problems. *Publication CRT-96-36, Centre de recherche sur les transports, Université de Montréal, Montréal, Canada*, 1996.

[9] T. G. Crainic, M. Toulouse, and M. Gendreau. Towards a taxonomy of parallel tabu search heuristics. *INFORMS Journal of Computing*, 9(1):61–72, 1997.

[10] E. Taillard. Some efficient heuristic methods for the flow shop sequencing problem. *European Journal of Operational Research*, 417:65–74, 1990.

[11] Bruno-Laurent Garica, Jean-Yves Potvin, and Jean-Marc Rousseau. A parallel implementation of the tabu search heuristic for vehicle routing problems with time window constraints. *Computers & Operations Research*, 21(9):1025–1033, November 1994.

[12] I. De Falco, R. Del Balio, E. Tarantino, and R. Vaccaro. Improving search by incorporating evolution principles in parallel tabu search. In *Proc. of the first IEEE Conference on Evolutionary Computation-ICEC'94*, pages 823–828, June 1994.

[13] S. Nair and A. Freville. A parallel tabu search algorithm for the 0-1 multidimentional knapsack problem. *11th International Parallel Processing Symposium*, April 1997.

[14] H. Mori and T. Hayashim. New parallel tabu search for voltage and reactive power control in power systems. *In Proc. of the 1998 IEEE International Symposium on Circuits and Systems - ISCAS'98*, pages 431–434, May 1998.

[15] Sadiq M. Sait, Habib Youssef, and Mahmood R. Minhas Aiman El-Maleh. Iterative heuristics for multiobjective vlsi standard cell placement. *Proceedings of IJCNN'01, International Joint Conference on Neural Networks*, 3:2224–2229, July 2001.

[16] Ronald R. Yager. On ordered weighted averaging aggregation operators in multicriteria decision making. *IEEE Transaction on Systems, MAN, and Cybernetics*, 18(1), January 1988.

[17] Sadiq M. Sait, Mahmood R. Minhas, and Junaid A. Khan. Performance and low-power driven VLSI standard cell placement using tabu search. *Proceedings of the 2002 Congress on Evolutionary Computation*, 1:372–377, May 2002.